Линк към хранилище в Github: <https://github.com/danisini/RasterGraphics>

**Проект 17: Растерна графика**

1. **Увод**

**Описание и идея на проекта**

Разглежданият проект реализира конзолно приложение, което представлява конзолен редактор на растерни изображения. Той поддържа работа с формати PPM, PGM и PBM. Може да стартира сесия/и, прилага различни трансформации върху изображенията и записва резултата.

**Цели и задачи на разработката**

* Подходящо представяне на снимките и сесията.
* Подходящо представяне и на системата, която ще обработва данните.
* Изграждане на леснодостъпно меню.
* Реализиране на основни функционалости на проекта ( отваряне, затваряне, записване, помощно меню, изход от програмата)
* Реализиране на допълнителни фунцкионалости на проекта (посивяване на снимката, почерняване на снимката, обръщане на цветовете в снимката, създаване на колаж и др.)
* Опция за записване на промените само при написването на команди save/ save as

**Структура на документацията**

Документацията е структуриране в следните основни точки:

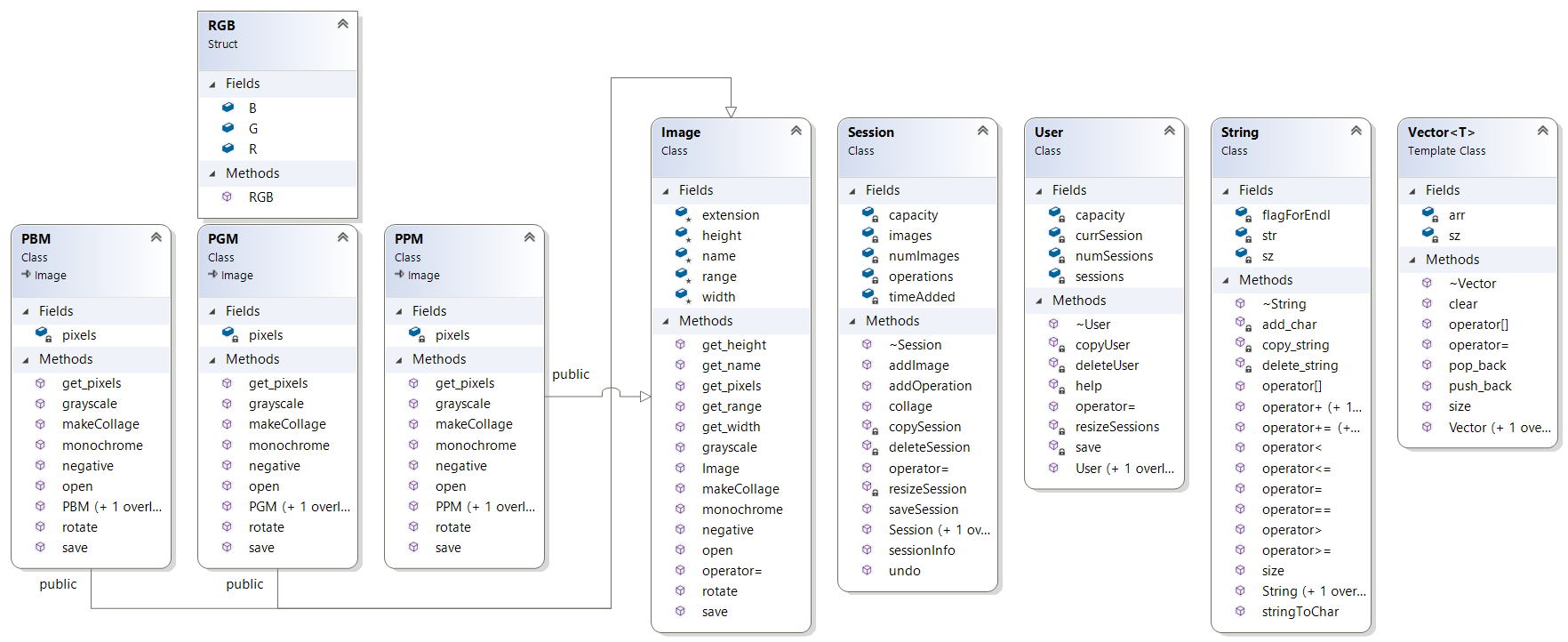
1. Увод
2. Проектиране
3. Реализация
4. **Проектиране**

По време на работата се стреми да се следват добрите принципи и практики на ООП дизайна. Реализирани са следните 7 класа:

* PPM
* PGM
* PBM
* Image
* Session
* User
* String
* Vector <T>

Реализирани са допълнителните класове String и Vector<T> с цел избягаване на допълнителна работа с динамична памет.

Те имат следната йерархия:



* **Клас Vector<T>**: темплейтен клас, реализиращ динамичен масив

**Private** член-данни:

* char\* arr; - указател към масив от тип Т
* size\_t sz; - големина на масива

**Public** член-функции:

* Vector(); - конструктор по подразбиране
* Vector(const Vector<T>&) = delete; - няма нужда от копиращ конструктор
* Vector<T>& operator=(const Vector<T>&) = delete; - няма нужда от оператор=
* ~Vector(); - деструктор
* void push\_back(const T&); - функция за добавяне на нов елемент от тип Т
* void pop\_back(); - функция за премахване на последния добавен елемент
* T& operator[](const size\_t&)const; - функция, достъпваща пряко елемент от масива по подаден индекс
* size\_t size()const; - функция, за връщане големината на масива
* void clear(); - фукнция за изтриване на цялата информация от масива
* **Клас String**: реализира динамичен масив от символи

**Private** член-данни:

* char\* str; - пойнтър към масив от символи
* size\_t sz; - големина на масива

**Private** член-функции:

* void delete\_string();
* void add\_char(const char);

**Public** член-функции:

* String(const char\* = ""); - конструктор
* String(const String&); - копиращ конструктор
* String& operator=(const String&); - оператор равно
* ~String(); - деструктор
* size\_t size()const; - метод за връщане на големината на масива
* bool operator==(const String&)const; - оператор ==
* bool operator<(const String&)const; - оператор <
* String operator+(const String&)const; - опеартор +, добавящ стринг
* String operator+(const char&)const; - опеартор +, добавящ стринг
* String& operator+=(const String&); - опеартор +=, добавящ стринг
* String& operator+=(const char&); - опеартор +=, добавящ стринг
* char& operator[](const size\_t&); - осигурява пряк достъп до елемент на масива
* bool operator<=(const String&)const; - оператор <=
* bool operator>=(const String&)const; - оператор >=
* bool operator>(const String&)const; - опеартор >
* const char\* stringToChar(){return str;} – преобразуваща член-фунцкия от String към char\*;
* friend std::istream& operator>>(std::istream& is, String& str); - приятелска функция за въвеждане на String;
* friend std::ostream& operator<<(std::ostream& os, const String& str); - приятелска функция за извеждане на String
* friend std::istream& getline(std::istream& is, String& str); - приятелска функция за въвеждане на String до въвеждане на нов ред
* **Клас PPM**: реализира снимки с разширение .ppm. Наследява **Image**

**Private** член-данни:

* Vector <RGB> pixels; // динамичен масив от стойностите на пикселите на снимката

**Public** член-функции:

* PPM(); - конструктор по подразбиране
* PPM(String); - конструктор с параметът името на файла
* virtual void open() override; - функция за отваряне на файла и четене на данните му
* virtual void save(const String&); - функция записваща промените в снимката
* virtual void grayscale(); - функция, която преобразува снимката до нюанси на сивото
* virtual void monochrome(); - функция, която прави снимката черно-бяла
* virtual void negative(); - функция, която прави негатив(цветово обръщане) на изображението
* virtual void rotate(const String&); - функция, която завърта снимката на 90 градуса или наляво, или надясно в зависимост от подадения параметър
* virtual const RGB get\_pixels(const size\_t& ind)const{return pixels[ind];} – функция, която връща стойността на пиксел с пореден номер ind от изображението
* virtual void makeCollage(Image\*, Image\*, const String&); - функция, създаваща колаж от две снимки с ориентация или хоризонтална, или вертикална
* **Клас PGM**: реализира снимки с разширение .pgm. Наследява **Image**

**Private** член-данни:

* Vector <size\_t> pixels; // динамичен масив от стойностите на пикселите на снимката

**Public** член-функции:

* PGM(); - конструктор по подразбиране
* PGM(String); - конструктор с параметът името на файла
* virtual void open() override; - функция за отваряне на файла и четене на данните му
* virtual void save(const String&); - функция записваща промените в снимката
* virtual void grayscale(); - Заради полиморфизма присъства като член-функция, въпреки че няма никаква реализация.
* virtual void monochrome(); - функция, която прави снимката черно-бяла
* virtual void negative(); - функция, която прави негатив(цветово обръщане) на изображението
* virtual void rotate(const String&); - функция, която завърта снимката на 90 градуса или наляво, или надясно в зависимост от подадения параметър
* virtual const RGB get\_pixels(const size\_t& ind)const{RGB curr;curr.R = curr.G = curr.B = pixels[ind];return curr;} – функция, която връща стойността на пиксел с пореден номер ind от изображението, като, тъй като трябва да върне RGB стойност, то на трите полета R, G и B се присвоява еднаква стойност.
* virtual void makeCollage(Image\*, Image\*, const String&); - функция, създаваща колаж от две снимки с ориентация или хоризонтална, или вертикална
* **Клас PBM**: реализира снимки с разширение .pbm. Наследява **Image**

**Private** член-данни:

* Vector <bool> pixels; // динамичен масив от стойностите на пикселите на снимката

**Public** член-функции:

* PBM(); - конструктор по подразбиране
* PBM(String); - конструктор с параметът името на файла
* virtual void open() override; - функция за отваряне на файла и четене на данните му
* virtual void save(const String&); - функция записваща промените в снимката
* virtual void grayscale(); - Заради полиморфизма присъства като член-функция, въпреки че няма никаква реализация.
* virtual void monochrome(); - Заради полиморфизма присъства като член-функция, въпреки че няма никаква реализация.
* virtual void negative(); - функция, която прави негатив(цветово обръщане) на изображението
* virtual void rotate(const String&); - функция, която завърта снимката на 90 градуса или наляво, или надясно в зависимост от подадения параметър
* virtual const RGB get\_pixels(const size\_t& ind)const{RGB curr;curr.R = curr.G = curr.B = pixels[ind];return curr;} – функция, която връща стойността на пиксел с пореден номер ind от изображението, като, тъй като трябва да върне RGB стойност, то на трите полета R, G и B се присвоява еднаква стойност.
* virtual void makeCollage(Image\*, Image\*, const String&); - функция, създаваща колаж от две снимки с ориентация или хоризонтална, или вертикална
* **Клас Image**: абстрактен клас.

**Protected** член-данни:

* char extension[3]; // разширение
* String name; // име на изображението
* size\_t height, width, range; // височина, широчина и цветови диапазон на изображението

**Public** член-функции:

* Image(){;} – конструктор по подразбиране
* Image& operator=(const Image&); - оператор=
* virtual void open() = 0; - *pure virtual* функция за отваряне на файлове
* const String get\_name()const {return name;} – връща името на изображението
* virtual void grayscale() = 0; - *pure virtual* функция за преобразуване на цветовете на изображението до нюанси на сивото
* virtual void monochrome() = 0; - *pure virtual* функция за преобразуване на цветовете на изображението до черно и бяло
* virtual void negative() = 0; - *pure virtual* функция за обръщане на цветовете
* virtual void save(const String&) = 0; - *pure virtual* функция за записване на новите данни за изображението
* virtual void rotate(const String&) = 0; - *pure virtual* функция за завъртане на изображението или наляво, или надясно
* virtual const size\_t get\_range()const{return range;}- функция, връщаща цветовия диапазон на изображението
* virtual const RGB get\_pixels(const size\_t&)const = 0; - *pure virtual* функция, връщаща пиксел по зададен пореден номер
* const size\_t get\_width()const{return width;} – фукция, връщаща широчината на изображението
* const size\_t get\_height()const{return height;} – функция, връщаща височината на изображението
* virtual void makeCollage(Image\*, Image\*, const String&) = 0; - *pure virtual* функция, създаваща колаж от две снимки с еднакво разширение и размерности, като има два варианта – с хоризонтална или с вертикална ориентация
* **Клас Session**: реализира сесиите на приложението

**Private** член-данни:

* Image\*\* images; - указател към масив от указатели към обекти от клас Images
* Vector <size\_t> timeAdded; - динамичен масив с информация кога е била добавена дадена операция спрямо броят снимки до момента
* Vector <String> operations; - динамичен масив с операциите, които ще бъдат приложени при запазване на файловете
* size\_t numImages, capacity; - брой изображение, капацитет на изображенията, който ще се увеличава с 8 всеки път, когато броят на снимките достигне 8, за да се избегне преоразмеряване на масива на всяка нова снимка.

**Private** член-функции:

* void deleteSession(); - с цел спазване на добрите практики на ООП функция за триене на обекти
* void copySession(const Session&); - с цел спазване на добрите практики на ООП функция за копиране на обекти
* void resizeSession(); - функция за преоразмеряване на масива от обекти

**Public** член-функции:

* Session(); - конструктор по подразбиране
* Session(const Session&); - копиращ конструктор
* Session& operator=(const Session&); - оператор за присвояване
* ~Session(); - деструктор
* void addOperation(const String&); - функция за добавяне на операция върху снимките от текущата сесия
* void undo(); - функция за премахване на последната операция от текущата сесия, ако има такава
* void addImage(String path); - функция за добавяне на снимка към текущата сесия
* void sessionInfo(const size\_t&); - функция за извеждане на информация за текущата сесия, обвързана с наличните снимки и операциите върху тях
* void saveSession(const String&); - функция за запазване на информацията от сесията
* void collage(const String&, String, String, String); - функция за създаване на колаж на две снимки с определената ориентация и по зададено име за изображението, където колажът евентуално ще се запише( ако потребителят реши)
* **Клас User**: реализира менюто на приложението

**Private** член-данни:

* Session\*\* sessions; // указател към масив от указатели от тип Session
* size\_t numSessions, capacity, currSession; // брой сесии, капацитет, и текуща сесия

**Private** член-функции:

* void deleteUser(); - с цел спазване на добрите практики на ООП това е функция за триене на обекти
* void copyUser(const User&); - с цел спазване на добрите практики на ООП това е функция за копиране на обекти
* void resizeSessions(); - функция за преоразмеряване на масива от указатели
* void save(const String&); - функция за записване на текущата сесия
* void help(); - функция за извеждане на командите, техния синтаксис и предназначение

**Public** член-функции – Реализирана е единствено голямата четворка

1. **Реализация и уточнения**

**Още уточнения:**

* Класът String въвежда данните символ по символ
* Използвах формула от интернет за правене на изображението в сиви цветове. Формулата я взех от този сайт: <https://rkenmi.com/posts/ppm-image-to-grayscale-c>
* using namespace std се изпозлва само в main.cpp с цел спазване на добрите практики.
* Допълнителните методи за копиране и триене са също с цел спазване на добрите практики.
* Освен това на много места параметрите на функциите са константни и по референция отново с цел спазване на добрите практики.
* Командата *close* затваря всички отворени сесии.
* Командата save as се въвежда като saveas с цел по-чист код, тъй като в другия проект е направена функция за броене на интервали, която помага за работата на менюто, ако е възможен интервал между *save* и *as.*
* Има по един интервал разстояние между въвежданите думи. Няма излишни интервали в началото и в края на командите.
* Команда *load* влече след себе си името на само една снимка с цел по-чист код отново.
* Приема се, че колаж се създава в изцяло нов файл.